DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-386

Automated Lane Detection by K-means Clustering: A Machine Learning Approach

Ajaykumar R, Arpit Gupta, Prof S N Merchant, Indian Institute of Technology Bombay, Mumbai

Abstract

With the advent of the driverless cars, the importance and accuracy of lane detection has achieved paramount importance in the field of perception and imaging. In this paper, we propose an algorithm to achieve lane detection on roads using the real-time data gathered by the camera and applying K-means clustering method to report data in a manner suitable to create a solvable map. The proposed method uses the physical nature of the data to cluster the data. Silhouette coefficient is used to determine the number of clusters in which the data should be divided. Lanes are interpolated to get the correct markings. We demonstrate the efficacy of, the proposed method using real-time traffic data to noise, shadows, and illumination variations in the captured road images, and its applicability to both marked and unmarked roads.

1. Introduction

With the start of the new millennia, driverless cars have achieved great impetus due to DARPA Grand Challenge and now-a-days, many automobile companies like Mercedes, Audi and Nissan have entered in the field of driverless technology. Any driverless technology for an automobile needs to sense the environment and access data that are vital for taking real time decisions to control the car. This is achieved by the help of various types of sensors such as Camera, Light Detection and Ranging (LiDAR) sensors, Radio Detection and Ranging (RADAR) sensors [2], Ultrasonic sensors, Stereo vision camera sensors, even thermal imaging in certain cases. Among all of these sensors, the camera sensors are one of the cheapest and most readily available sensors. It is also the kind of sensor whose data is very easy to visualize. Thus Computer Vision is one of the core aspects for any autonomous technology that controls any automobile for the foreseeable future.

An important part of autonomous driving is lane detection. Lanes provide an order in urban driving scenarios. They provide a direction for the car to follow. It is of utmost importance for a vehicle to drive in a lane and judge when to cross the lane for the purpose of overtaking or taking turns. The paper deals with improving the result of Lane Detection with a Machine Learning Approach. This improves the accuracy of lane detection to a higher degree than traditional lane detection algorithms. The algorithm's accuracy is limited by the quality of the image taken and is of polynomial time complexity.

The problem statement is to improve current lane detection algorithms. Our method involves state-of-the-art and proven machine learning algorithms. Traditionally, B-snake curve fitting is used for this purpose. There are a number of shortcomings with B-snake algorithm. It uses vanishing point method to extrapolate the line segment which represents lane data. This does not work well at the truncation points of the images. It also fails when the boundaries in the image are not clear. The proposed algorithm takes care of these shortcomings as it does not depend on the perspective transform and does the lane detection on original image itself.

We propose K-Means Clustering to obtain a reasonable estimate of the number of clusters in the data. That is, we can cluster the data accurately into reasonable number of clusters. Compared to other lane detection algorithms like B-spline, this is a very simplified model which does not require any calibration for perspective transformations as in B-spline. The present algorithm takes the raw image and outputs the lane data without change in the kernel of view space.

2. Background

2.1. Shortcomings of existing methods in Lane Detection

In B-snake algorithm for lane detection [1], the following is a short description of the algorithm:

1. Construct an edge map of image using Canny Edge Detector (according to CHEVP model)
2. Divide the image into a number of segments for compensating the bend of the lane
3. Apply Hough Transform on each of the image segments and individually detect the vanishing points of each segment and further detect the horizon of the image
4. From these vanishing points, compute the control points for the construction of the B-snake
5. From these control points, a spline would then have to be constructed. From this, the lane boundary would be extrapolated using the width of the road

Following are the issues with the B-snake algorithm:

1. In our understanding of the entire algorithm, it is clearly capable of detecting the lane if there was only a single lane in the image. It is not clear how multiple lanes would be detected and/or distinguished from each other.
2. The CHEVP model (Canny/Hough Estimation of Vanishing Points) does not remove the illumination noises to the required levels. There are too many cases where this method of calculating the vanishing point fails mainly because of illumination noise. This is because, in certain cases, the noise is sharper, distinguishable than the lane part of the image.
3. Vanishing points may not be calculated for certain images if the entire lane is not visible in the image. Thus we feel that it is not right to focus attention on finding the mid-lane. Rather, we must detect each lane boundary individually. This is important for autonomous systems because it focuses on a wider range of possibilities for which lane detection is attempted. In case of corrupted output as seen in case of non-uniform illumination in the image, we may assume that the

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-386

previous lane boundary is parallel to the present lane boundary

2.2. Advantages of using present algorithm

The major advantage of this algorithm is that it does not assume that the lanes that are to be detected are of any prescribed shape or number. It is capable of detecting lanes of any number, width and shape even if the shape changes dynamically. This has been made possible because of the use of an unguided machine learning algorithm to detect all the lanes and its number. Thus we don't see a necessity of assuming that the lane boundaries are parallel to each other. The algorithm also detects multiple lanes as viewed in the image.

A large fraction of lane detection algorithms including this one may not produce reliable results if there is a drastic change in illumination [7] in the image which may be caused in certain regions of the image due to reflected rays from the sun that are incident directly on the lenses of the camera used. This algorithm does capture lanes in all cases, but has a tendency to classify these bright spots of sunlight also as lanes. This case is not handled by this algorithm and may be specially treated after applying an illumination correction on the image. The algorithm produces the best results when the lane is clearly marked on a cloudy day with negligible amount of sunlight.

3. Algorithm Steps:

We overcome the above shortcomings of the B-Snake algorithm in the following way:

1. We assume that the camera is always pointed towards the road i.e. the road is a major part of the image. By applying a 3-channel filter, we can easily get the lane boundaries from the image as a binary threshold image
2. From the binary image, extract all contours and reject all small area contours. This removes the salt and pepper noises from the image
3. Apply Probabilistic Hough Transform (PHT) to the binary image. We end up with an output that has a lot of clustered lines in the region of the lane boundaries. In simpler words, the lines are clustered at the lane boundaries
4. Then we segregate the output from the PHT according to which contour they are present in i.e. group the line segments according to the contours they occur in
5. Apply K-Means Clustering algorithm to each group of line segments. The output would give us an idea about the shape of the lane. The cluster means inside each group would be the required output for the lane boundaries in the image
6. Arrange the means in a particular order to plot the spline representing the lane boundary

This algorithm has an advantage that it can find all lanes in the image with a high degree of accuracy. An even better observation is that the shape of the lane plays absolutely no role in its detection. Thus no lane model is assumed for its detection. Also as a result, any number of lanes may be simultaneously detected in the image.

This represents a generalized algorithm for lane detection from a camera and it gives the best results when there are no sun spots or areas of bright color on the road. As for illumination correction, there are many algorithms available for removing those bright regions and the output of such a correction must be fed to this algorithm for optimum results. The next section talks of the algorithm in detail.

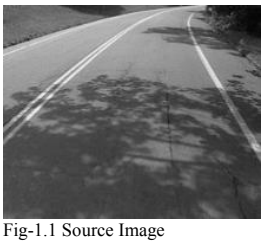
Fig-1.1 Source Image


Fig-1.2 3-Channel Thresholding


Fig-1.3 Area Thresholding

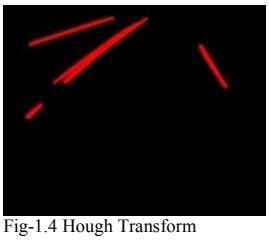
Fig-1.4 Hough Transform

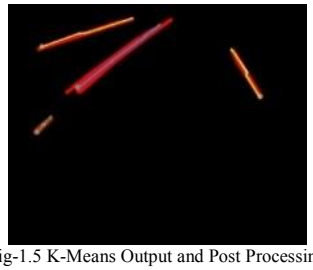
Fig-1.5 K-Means Output and Post Processing

3.1. Preprocessing Image

3.1.1. White Color Thresholding

A basic assumption of this algorithm is that any white patch on the road is a lane marking. Then this white patch is captured by a 3-channel filter with an appropriate threshold.

For each pixel, individual threshold are applied to 3-channels, red, green and blue channel. For an 8-bit image, the green threshold is set as 200, the red threshold is set as 200 and the blue threshold is set as 200.

$$\forall p \in Image\ I, \langle R|G|B\rangle_p > \langle 200|200|200\rangle \Rightarrow \langle R|G|B\rangle_p \equiv \langle 255|255|255\rangle$$

3.1.2. Area Thresholding

After white color thresholding, salt and paper noises tend to remain in the image. To remove them, the white color patches are assigned as separate closed contours. These contours are then further passed through an area filter to reject all oddly placed white markings. All contours with the area lesser than 100 sq.pixel (for image resolution of 1280×720) are rejected. Thus, we obtain an image which has the lane markings in a black background. This removes any oddly illuminated isolated pixel or a small group of pixels.

3.1.3. Hough Transform

Now we have obtained lane markings from the image, this does not specify the lane boundaries. Interpretation of this line segment data is required for getting the lane boundary position. We need to accurately determine the boundary of multiple lanes.

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-386

Thus we apply Probabilistic Hough Transform [4][5] on the resultant image. This gives us a set of line segments which are useful to determine the lane boundary position.

Probabilistic Hough Transform is later applied on the resulting binary image that is obtained from the 3 channel filter and area threshold. This provides us with a vector of line segments on which the clustering algorithm is applied. K-means clustering was chosen as it provided a natural principle to go about for lane detection. Hough Transform does give us the lane marking position, but it does the same in an unclustered and unusable output. The output of the probabilistic hough line transform comes in the form of extremes of the line. Let each line be given an index *i*. Initial point be denoted by index o and final point be denoted by index *f*. So extreme points are $(x_{oi}\ y_{oi}\ x_{fi}\ y_{fi})$

After we apply hough transform, we then split the line segments into groups corresponding to the nearest white patch. This was done by enclosing a tight rectangle (close fit) around each contour. Let each contour be defined by *(k)*. So a line (*i)* with initial point (*o)* and final point (*f*) in a contour (*k*) is denoted by $(x_{oi}^k\ y_{oi}^k\ x_{fi}^k\ y_{fi}^k)$. If the midpoint of the line lies within a rectangle, then it is said to be associated with the group of lines corresponding to that white patch.

Any line segment that lies between two patches is classified as noise and rejected from the set of line segments. This also aids us in decreasing the runtime of the algorithm.

Hough transform was chosen as the line segments obtained after the white thresholding filters are almost always concentrated on the lane marking positions. This aggregation of line segments tells us that the LOCALISED MEANS of these line segments may be considered as the lane marking positions.

K-Means was chosen to be a solution for this unique problem. It can be used very efficiently to extract these localized means provided we know how many such means are there.

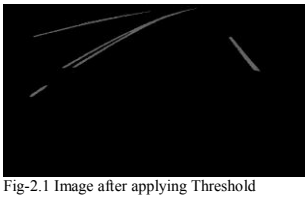
Fig-2.1 Image after applying Threshold

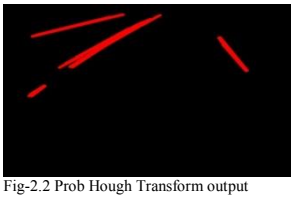
Fig-2.2 Prob Hough Transform output

3.2. Application of K-means to get Line Data

3.2.1. Procedure to Apply the Clustering Algorithm

We intend to apply the clustering algorithm inside each group of line segments (split according to white patches). The basic idea of clustering Line segments means to find the "Average" of all the line segments inside a cluster.

Let us take the case of point data. K-Means clustering of point data involves finding clusters of point data. Thus the algorithm provides us with the means of the clusters.

Within a rectangle, we arrange all the indices of the hough lines such that $(y_{oi}^k > y_{fi}^k)$ so that the line segments can be interpreted in a progressive fashion only as per the distance from the vehicle. Now all the observations within a rectangle have these two properties:

- $C_1 \cup C_2 \cup C_3 \cup \ldots \cup C_K = \{1,2,3,\ldots,n\}$. In other words, all the line segments are covered within a rectangle.
- $C_k \cap C_{k'} = \varphi$ for all $k \neq k'$. In other words, the clusters are non–overlapping.

If the data has **n** number of clusters, then the algorithm for obtaining **n** clusters is as follows:
(Sample Data with 4 clusters, K-Means Terminates here in 6 iterations)

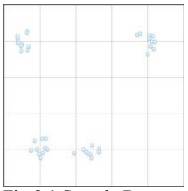
Fig-3.1 Sample Data

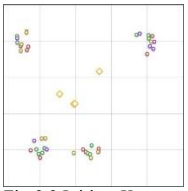
Fig-3.2 Initiate K-Means

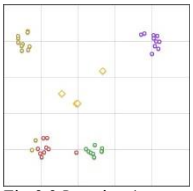
Fig-3.3 Iteration 1

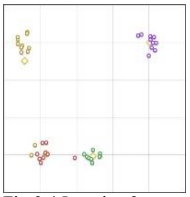
Fig-3.4 Iteration 2

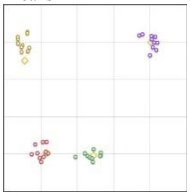
Fig-3.5 Iteration 3

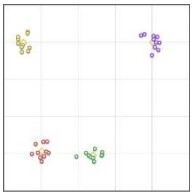
Fig-3.6 Iteration 4

A - Initialization: Randomly associate **n** line segments as the cluster means, c_{m_j}

B - Iteration: Find the distance of each point from all the cluster means. The distance is calculated using Euclidean distance and associates each data point with appropriate cluster. The distance between the line segments is termed to be the distance between their midpoints. Distance between two lines l_1 & l_2 with mid points (x_{ml_1}, y_{ml_1}) and (x_{ml_2}, y_{ml_2}) is defined as

$$dist(l_1, l_2) = \sqrt{\left(x_{ml_1} - x_{ml_2}\right)^2 + \left(y_{ml_1} - y_{ml_2}\right)^2} \quad (1)$$

For clustering the line in a particular cluster, C_k, the following criteria is used

$$k = \ _{j=1}^{n}min\left(dist\left(i, c_{m_j}\right)\right) => i \in C_k. \quad (2)$$

C - After Updating: Calculate the new cluster means which are the means of the clusters so formed. Average is calculated as the centroidal line segment defined as the line segment joining the centroid of all the initial points (x_{oi}, y_{oi}) and centroid of all the final points (x_{fi}, y_{fi})

$$c_{m_k} = avg\{i: i \in C_k\} \quad (3)$$

D - Termination: Check the position of the new means with the old ones. If they are "very close" then terminate the process, or else go to step 2 with the updated means as the cluster means.

This process is very effective in determining the clusters from the data. To figure out the number of clusters in the data, again an iterative approach is used. This is accomplished by looking at a coefficient called as the Silhouette Coefficient. After clustering the lines, the mid-point of the line is used to determine the Silhouette Coefficient.

E - Silhouette Coefficient:

Within a cluster, for each point:

- a_{i_k} is defined as the average distance of the point from the other points in the same cluster, (cohesion factor)

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-386

$$a_{i_k} = avg\{dist(i,j): i \neq j, j \in C_k\} \quad (4)$$

- b_{i_k} is defined as the average distance of the point from the other points in the nearest cluster, (separation factor)

$$b_{i_k} = \min(dist(i,j): i \in C_k, \quad j \in C_p \quad (5)$$

$$p = \ _{t=1}^{n}\min(dist(c_{m_k}, c_{m_t}),\ t \neq k))$$

The silhouette coefficient for that data point is then defined as:

$$s_i = \frac{(b_{i_k} - a_{i_k})}{\max\left((b_{i_k},\ a_{i_k})\right)} \quad (6)$$

Then the overall silhouette coefficient is defined as:

$$S = \ _{\forall i} avg(s_i) \quad (7)$$

From the definition of the silhouette coefficient, it is clear that the range of the coefficient is between -1 and 1.

So we have the following,

$$s_i = \begin{cases} 1 - \frac{a_{i_k}}{b_{i_k}} & if\ b_{i_k} > a_{i_k} \\ 0 & if\ b_{i_k} = a_{i_k} \\ \frac{b_{i_k}}{a_{i_k}} - 1 & if\ b_{i_k} < a_{i_k} \end{cases} \quad (8)$$

$$(\ a_{i_k} \ll b_{i_k} => s_i \to 1,\ a_{i_k} \gg b_{i_k} => s_i \to -1)$$

In the given data, if variation within the cluster is small (a_{i_k} is small), and variation from data points which are part of another cluster (b_{i_k} is large), this indicates a good clustering of data. (Low cohesion and large separation)

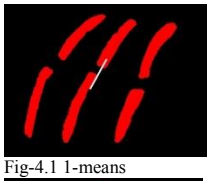

Fig-4.1 1-means

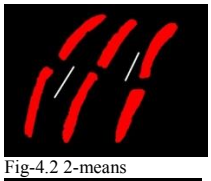

Fig-4.2 2-means

Fig-4.3 3-means

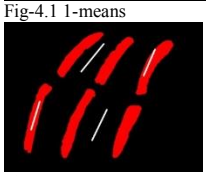

Fig-4.4 4-means

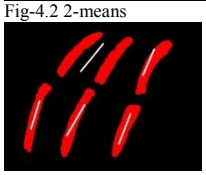

Fig-4.5 5-means

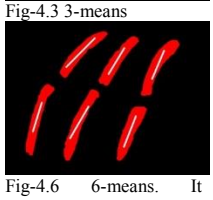

Fig-4.6 6-means. It represents the data perfectly

Fig-4.7 7-means. Data clustered to 7 clusters. This is not an appropriate number of clusters in the data. So the algorithm decides that the earlier method to cluster the data was appropriate and terminates the loop

Figures positioned above Fig-4.1 to Fig-4.6 show progression of means from 1 to 6 in a clearly clustered data. Algorithm terminates after calculating coefficient for 7 means as it is smaller than that of 6 means.

Then we can state that if we divide the data set into partitions such that the average silhouette coefficient of the entire data set or the average silhouette coefficient is maximized, then the data is adequately clustered into the correct number of clusters. In this case, the silhouette coefficient approaches 1 (Any value more than 0.7 indicates good clustering).

Also note that if the data has not been clustered properly or if the data has no clear number of clusters, the Silhouette Coefficient is smaller compared to that of a properly clustered data. This happens because the data has cohesion and separation in comparable ranges. This indicates that some sample data have been misclassified.

K-means clustering fails when the data distribution is uniform i.e. there is no clear clustering of data. In these cases, silhouette coefficient comes near 0. (Any value between 0 and 0.25 means the data is uniformly distributed)

If the Silhouette Coefficient [6] is positive and close to 1, say something like 0.75, from above definition of Silhouette coefficient, we have the inequality $a_{i_k} < b_{i_k}$. This means that the factor of cohesion is small compared to factor of separation. This means that on an average, distance between data points within the same cluster is smaller than the distance between points in different clusters. This means that points within the same cluster are close to each other compared to points in different clusters.

This is indicative of good clustering. Thus Silhouette Coefficient value close to 1 indicates good clustering of data.

On the other hand, if the Silhouette Coefficient is small, say something like less than 0.25, it means that the distance between points within the same cluster I comparable to points in different clusters. This means that the data is not clustered properly. Better clustering is possible. An immediate solution is to increase the number of clusters to be considered for clustering the data. This may or may not increase the Silhouette Coefficient.

This may occur because increasing the number of clusters may lead to better clustering, but beyond a point, the clusters start to overlap. If we keep on blindly increasing the clusters, then we get to a situation where each and every single data point is a cluster on its own. This leads to breakdown of the concept of cluster in data. Thus it is unwise to keep increasing the number of clusters if the silhouette coefficient begins to fall.

A better implementation of the algorithm is to compute the cluster means of all reasonably possible number of clusters in the data keeping a threshold in the value of the Silhouette Coefficient, figure out the global maxima of the Silhouette Coefficient value and then it corresponds to the appropriate number of clusters in the data. This process would be computationally expensive and may

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-386

induce lag in the real time lane detecting module. Thus the algorithm was simplified to terminate whenever the Silhouette Coefficient falls.

So the Silhouette Coefficient is an important factor in cluster formation. It is to be noted that the silhouette coefficient just gives us an indication of the clustering in the data. It gives us a guess on how many clusters exist in the data.

This is the basic logic that is iteratively applied properly figure out the appropriate clustering of the data. The idea is to repeatedly evaluate the Silhouette Coefficient for all iterations and stop the process when the coefficient starts to fall from the previous value. In the end, we are left with some line segments that are the means of the clusters in the data.

3.2.2. Advantages

The line segments from the Hough Transform are mostly clustered around the lane markings. Thus the local means of these clusters or the 'clustered means' can be thought to be the accurate positions of the lane markings. This kind of clustered data openly demands K-means clustering for accurate implementations. The problem with the normal implementation of K-means clustering is that we need to know how many clusters exist in the data. This is not available with us in case of lane detection. So we get around that particular problem by comparing the cluster means that we obtain from the K-means algorithm with the data set available with us. We compare the means with the data by a parameter known as Silhouette Coefficient. Maximizing the silhouette coefficient leads to appropriate clustering of data. This accurately gives us the number of clusters in the data.

Advantage of using K-Means is that it provides us with a remarkably accurate output consistently. The algorithm's error is actually dependent on the error of the available data (Error generated from Hough Transforms) which are quite accurate by their own right.

K-Means Clustering algorithm is the most basic algorithm in case of Unsupervised Machine Learning Algorithms. The basic crux of the algorithm is widely understood and respected within the world of Algorithms. Thus this provides us with a fool-proof way of obtaining the lane from the image.

Another realization from the algorithm is that it is real time. Thus processing power and memory requirement is also quite within the limits.

3.2.3. Post Processing of K-Means output:

A simple way to use the K-Means output is to simply consider the midpoints of the cluster-mean-line-segments. They are the control points with which the entire lane may be represented as a spline which can be constructed from these points. As stated earlier, curved lanes have more means than straight lanes. This means that any curvature can be easily captured by this algorithm. Thus highly curving lanes get more data for representation and straight lanes get lesser data for representation.

Thus the midpoints of the cluster-mean-line-segments are the lane data. Lanes are commonly visualized as splines. Now one obstacle remains. As a result of using a randomized clustering initiation, these data points are not listed in order. We have not yet visualized the lane in its complete form. It is imperative to understand that the points still may not plot a lane as per our requirements. We cannot plot this spline properly if the order of the points is incorrect. Thus to make the data in a more useful form, we need to arrange the points in an order that would be used for plotting a spline i.e. there exists a starting point for the lane and an ending point for it.

This problem was solved by a common observation. Lanes are usually characterized as long thin strips irrespective of color. The image contours that bound these lanes are usually long and thin polygons. These polygons usually have tips from where the lanes begin. Thus the lane data point closest to this tip is the starting point of the lane. These sharp "tips" or "corners" are most often close to the beginning point of the spline which accurately represents the lane. So among the set of points we have obtained from the K-Means clustering, we need to figure the point closest to this "tip". This would be the starting point of the spline.

After finding this point, we can iteratively find the next point to this one by finding the closest point among the leftovers. This leads to an algorithm:

- For a contour, approximate it to a polygon and find the point of sharpest turn (many known algorithms exist to do the job).
- Find the lane point that is closest to the tip of the polygon.
- Iteratively find the next point that is closest to the recently added point.
- Joining them in this order gives us the natural spline which accurately maps the lane.

These post processing operations are usually much quicker than the algorithm run time and thus post no issues regarding the time complexity.

4. Results and Discussions

4.1. Testing Results

The output of the algorithm is a set of points with which the lane may be reconstructed using splines. It is a very usable format as we get pixel-wise information on which part of the image actually contains the lane markings.

This is very useful as the same information can be used to mark the lanes in a grid map after inverse perspective mapping to clearly map the environment. This information is clearly independent of the lane width.

It can be clearly seen that the output is best observed when the image is of uniform illumination. The lane markings must be clearly distinguishable. One clearly notices the fact that the algorithm is capable of detecting multiple lanes simultaneously and gives an output in a very user friendly format.

It must be noted that the algorithm is extremely sensitive to sunlight and thus may not give the required output in case of excessive sunlight. Best results were seen on cloudy days and on days with minimal sunlight.

Output of the algorithm is an array of points in the image which correspond to lane boundary locations in the image. A sample is shown below:


Fig-7.1 Raw Image 3

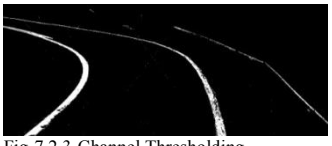
Fig-7.2 3-Channel Thresholding

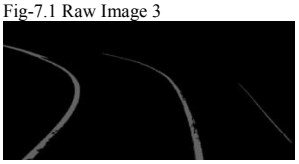
Fig-7.3 Area Thresholding

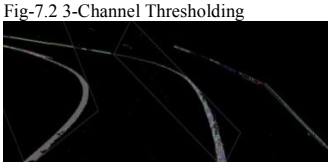
Fig-7.4 Marking Contours

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-386


Fig-7.5 Hough Transform


Fig-7.6 K-Means and post processing

Table 1 : Output of Raw Image 1

Lane 1:	Lane 2:
(48,212),(60,200), (71,187),(115,138), (134,125),(173,93), (197,79),(229,53), (268,26)	(49,74),(120,49), (128,48),(423,153), (399,120),(398,114), (378,88)

Array of points depicting the lane boundary positions by pixel positions in the image are shown above. This can further be used in Inverse Perspective Mapping to map the environment in a Grid Map [3]

Following images show the output of the lane detection algorithm:


Fig-5.1 Raw Image 1


Fig-6.1 Raw Image 2

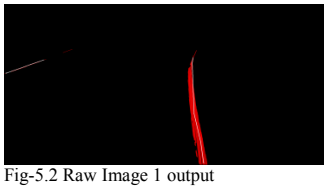
Fig-5.2 Raw Image 1 output


Fig-6.2 Raw Image 2 output


Fig-7.1 Sample Image 1


Fig-8.1 Sample Image 2

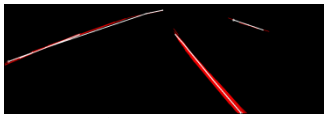
Fig-7.2 Sample Image 1 output


Fig-8.2 Sample Image 2 output


Fig-10.1 Sample Image 3


Fig-11.1 Sample Image 4

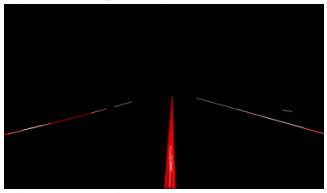
Fig-10.2 Sample Image 3 output

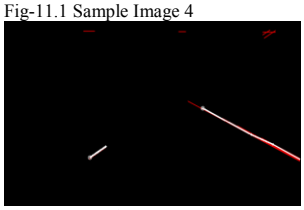
Fig-11.2 Sample Image 4 output

The next part talks of the computational complexity of the algorithm.

4.2. Computational Complexity

Let $m \times n$ be the image size with l number of lane markings.

1. 3 channel filter: $O(m \times n)$
2. Area Thresholding: $O(l)$
3. Hough Transform: $O(m \times n)$
4. K-Means on the Hough Output: $O(\sum(n \times k \times l)) = O(n \times k^2 \times l)$, $k \in \{2, 3, 4, \cdots, preferred\ limit\ on\ K-Means\}$, $l < m \times n$
   - Initiate number of means as 1.
   - Iteratively calculate k means by clustering.
   - After means have been calculated, find the silhouette coefficient of each data point as defined above.
   - Find average silhouette coefficient of the entire data set.
   - If the silhouette coefficient drops below the previous calculated value, present clustering is considered to be adequate. Else increase number of clusters.
5. Arranging output to plot a lane: $O(k \times \mathrm{k})$

On an average, the K-Means output converges to 4-5 clusters with the max number of clusters being the limit set (10 in our case). So the time complexity of the entire algorithm is polynomial, thus real time with a high degree of accuracy.

5. Conclusion and Future Developments

The algorithm was developed with the aim of detecting lane boundaries and converting their data into a usable format for further processing in autonomous systems. This aim has been clearly achieved by this algorithm and thus, can be integrated into autonomous automotive systems for various applications.

Future developments of the algorithm may include certain improvements in K-Means with better clustering approximations with a more mathematical basis, better illumination normalization techniques and better noise removal techniques.

5. References

[1] Wang Y, "*Lane detection and tracking using B-Snake*", Journal of Image and Vision Computing, pp. 269-280, 2003

[2] Peter Hyde, *Mapping forest structure for wildlife habitat analysis using multi-sensor*, Remote Sensing of Environment, Vol 102, Issues 1-2, May 2006, Pg 63-73

[3] Tao Yan, "*Seamless Stitching of Stereo Images for Generating Infinite Panoramas*"

[4] Pao D.C.W, Li H.F , Jayakumar R, "*Shapes Recognition Using the Straight Line Hough Transform: Theory and Generalization*", IEEE Transactions on Pattern Analysis & Machine Intelligence, vol.14, no. 11, pp. 1076-1089, November 1992, doi:10.1109/34.166622

[5] KIRYAT N, "*A Probabilistic Hough Transform*" Department of Electrical Engineering and Department of Computer Science, Technion, Israel Institute of Technology, Haifa 32000, Israel

[6] Peter J. Rousseeow, *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,* Journal of Computational and Applied Mathematics 20 (1987) 53-65 University of Fribourg, ISES, CH-I 700 Fribourg Switzerland

[7] Vijitha P, "*Better Quantification of Vitiligo by ICA on Illumination Corrected Images*", Medical Intelligence and Language Engineering (MILE) Laboratory, Department of Electrical Engineering Indian Institute of Science (IISc), Bangalore, India